“Atomic” Systems and Containers

Neal Gompa (Conan Kudo [ニール・ゴンパ])

Who am I?

- Professional technologist
- Humble maintainer of a handful of packages in the Fedora Project
- Diligent follower of the telecommunications industry
- Associate SQA Engineer at Datto, Inc

Contact Points:

- Twitter: @Det_Conan_Kudo
- Google+: +NealGompa

So… What are atomic systems?

Atomic systems are systems that are built and managed as indivisible units. Usually, there’s no visibility into the components that make up the system.

- For example, a live install of most Linux distributions would be an example of an “atomic” installation, as the software is pre-selected, pre-configured, and just applied onto the disk that way.

This is contrast with the traditional model, in which systems are highly componentized and each component is managed individually.

- For example, doing an update of individual packages or even installing/removing individual packages exposes the componentized nature of the system.

A review of how systems are built...

Today, Linux systems are split up into packages. These packages are stored in repositories (file trees on the Internet). Access to repositories are done through repository managers that call package managers to install, update, or remove sets of packages.

Why systems are built this way…

In the beginning, Linux systems were put together from sources of programs from various sites/servers, meaning that it was functionally impossible to construct Linux systems as singular units. Because they were disparate and often managed completely separately from distributions that collect and distribute code, it didn’t make sense to manage it in that way.

Now, distributions are often modifying and maintaining code separately from the projects the code was sourced from in order to maintain working systems. From a practical perspective, the distributions are usually now a centrally managed source of code/programs, as opposed to something that just grabs project code sources and builds them.

Why use atomic systems?

By having components managed “together” as if it is a singular unit, it is possible to create an absolutely reproducible system that can be heavily tested and hardened.

Why not use atomic systems?

Not everyone needs exactly the same software set, so it would be very difficult to satisfy everyone with a purely atomic system intended for general purpose use. Without a means to install new software, it becomes pretty useless.

Is there a good compromise?

Generally speaking, there’s a common set of functionality that needs to be present on every system.

Perhaps, if that core was atomically managed and there was a way to layer more things on top…

PROJECT

ATOMIC

Enter Project Atomic and Containers!

Project Atomic

Project Atomic is Red Hat’s approach to this particular problem. Essentially, a very minimal subset of components are composed into a system tree that are managed as a singular unit, and *containers* are used to add functionality.

Wait, what… Containers?

Containers are *extremely* minimal environments designed to run an application/service or a set of them with a (normally) *small* degree of isolation, usually just lack of awareness of the actual system.

Containers are typically used to set up instances of services that would normally manipulate large parts of the underlying system in a manner that is safe and distributable. Example implementations are Virtuozzo, LXC, Docker, and systemd-nspawn.

Hold on! Docker?!

So, Docker is a mechanism for constructing and maintaining containers of applications and services. Originally a wrapper around LXC, it now manages the necessary kernel functionality directly.

Most container implementations do not handle reproducible creation of containers. Docker does.

Project Atomic and Docker

Project Atomic utilizes Docker as a means to enable functionality to be layered on top of it. Without being able to reproducibly construct containers for software delivery, Atomic wouldn’t be useful as it is.

Demonstration

Using Atomic web UI (Cockpit) to use containers

Atomic + Containers

Atomic makes it possible to have your cake and eat it too, by using Docker as the foundation for extending the capability of the platform instead of using standard packages, while guaranteeing that the core of the OS is stable and safe.

Additionally, Atomic is SELinux-enabled to enable confinement, as containers don’t normally contain.

Wait, containers don’t contain?!

It seems counterintuitive, but containers don’t inherently contain anything. Essentially, containers are “self-contained” only in execution, not total system environment.

SELinux grants this security to containers by using its mechanism to enforce MAC (mandatory access control) to restrict what containers can do. Coupled with namespaces (another kernel feature that isolates environments), containers can actually contain!

So… how do I get Atomic?

You can get a “host image” for Atomic at the Project Atomic download page.

The Atomic Host is available in two freely available flavors:

- Fedora Atomic Host (derived from the latest Fedora code)
- CentOS Atomic Host (derived from Red Hat Enterprise Linux Atomic Host)

Additionally, there is an enterprise-grade supported option from Red Hat:

- Red Hat Enterprise Linux Atomic Host, which is marketed as a part of a system called the *Red Hat Atomic Enterprise Platform*

Sources and additional resources

- Red Hat Customer Portal: *Introduction to Linux Containers*: https://access.redhat.com/articles/1353593
- Daniel Walsh: *Container security: Do containers actually contain? Should you care?*
  - Presentation: https://fedorapeople.org/~dwalsh/Presentations/ContainerSecurity/#/
  - YouTube: https://www.youtube.com/watch?v=a9lE9Urr6AQ
- Daniel Walsh: *Super Privileged Containers*
  - Presentation: https://fedorapeople.org/~dwalsh/Presentations/SPC/#/
  - YouTube: https://www.youtube.com/watch?v=dM2Fc53Dtd4
- Adam Miller: *Immutable infrastructure, containers, & the future of microservices*: http://videos.cdn.redhat.com/summit2015/presentations/12017_immutable-infrastructure-containers-the-future-of-microservices.pdf
- Aditya Patawari: *Running your containers in a sane environment, Project Atomic*: https://www.youtube.com/watch?v=XCw5sViG2cw
- Project Atomic: *Introduction to Project Atomic*: http://www.projectatomic.io/docs/introduction/
- Matthew Micene: *Running Cockpit as a service in Fedora 22 Atomic Host*: http://www.projectatomic.io/blog/2015/06/running-cockpit-as-a-service/
- Jason Brooks: *Running a Containerized Cockpit UI from Cloud-init*: http://www.projectatomic.io/blog/2015/08/running-a-containerized-cockpit-ui-from-cloud-init/
- Project Atomic: http://www.projectatomic.io/
- Project Atomic: *Nulecule Specification*: http://www.projectatomic.io/docs/nulecule/
- Project Atomic: *Atomic App*: http://www.projectatomic.io/docs/atomicapp/
- Fedora Atomic Host: https://getfedora.org/cloud/download/atomic.html
- CentOS Atomic Host: https://wiki.centos.org/SpecialInterestGroup/Atomic/Download/
- Red Hat Atomic Enterprise Platform: https://access.redhat.com/products/red-hat-atomic-enterprise-platform

The End

Any Questions?